Project Management Using Point Graphs

Abbas K. Zaidi
Mashhood Ishaque
Alexander H. Levis

Conference on Systems Engineering Research
March 2007
Finish-Start Barrier

In CPM/PERT temporal constraints are allowed only between the end of an activity and start of the following activity. Thus could not represent nominally parallel activities.
Example

Constraints

Interval  A, B, C, D, E
Length [A] = 5
Length [B] = 5
Length [C] = 5
Length [D] = 2
Length [E] = 10

{ A Meets B
C Meets D
C Precedes B
}  

{ eE Precedes eD

Representation

?
Point Graphs

We propose a graph-based representation called Point Graphs that overcomes the shortcomings of traditional approaches to project management.
Outline

• Point Graphs
• Representation of Constraints
• Operations on Point Graphs
• Scheduling Algorithms
Point Graphs

Node = point on the timeline.

Edge = temporal relation between points.

All LT Edges must have lengths specified for scheduling algorithms to work.

Knowledge Representation for Point-Interval Temporal Logic.

Inference using graph searches.

Node P1, P2, P3

Point P1, P2, P3
P1 < P2
P2 ≤ P3
Stamp [P1] = 4500
Length [P1,P2] = 100
Examples of Constraints (Qualitative)

<table>
<thead>
<tr>
<th>Interval X, Y</th>
<th>$sX \rightarrow eX \quad sY \rightarrow eY$</th>
</tr>
</thead>
<tbody>
<tr>
<td>X Before Y</td>
<td>$sX \rightarrow eX \quad sY \rightarrow eY$</td>
</tr>
<tr>
<td>X Precedes Y</td>
<td>$sX \rightarrow eX \quad sY \rightarrow eY$</td>
</tr>
<tr>
<td>X Meets Y</td>
<td>$sX \rightarrow eX; sY \rightarrow eY$</td>
</tr>
</tbody>
</table>

... Interval to Interval Constraints

<table>
<thead>
<tr>
<th>Point P, Interval Y</th>
<th>$P \quad sY \rightarrow eY$</th>
</tr>
</thead>
<tbody>
<tr>
<td>P Before Y</td>
<td>$P \quad sY \rightarrow eY$</td>
</tr>
<tr>
<td>P Starts Y</td>
<td>$P; sY \rightarrow eY$</td>
</tr>
<tr>
<td>P During Y</td>
<td>$sY \rightarrow P \rightarrow eY$</td>
</tr>
<tr>
<td>P Finishes Y</td>
<td>$sY \rightarrow P; eY$</td>
</tr>
<tr>
<td>Y Before P</td>
<td>$sY \rightarrow eY \rightarrow P$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Point to Interval Constraints</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Point P, Q</th>
<th>$P \quad Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>P Before Q</td>
<td>$P \rightarrow Q$</td>
</tr>
<tr>
<td>P Equals Q</td>
<td>$P; Q$</td>
</tr>
<tr>
<td>P Precedes Q</td>
<td>$P \rightarrow Q$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Point to Point Constraints</th>
</tr>
</thead>
</table>

*System Architectures Laboratory*
Examples of Constraints (Quantitative)

<table>
<thead>
<tr>
<th>Stamp [P] = t</th>
<th>( t \rightarrow P )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length [X] = d</td>
<td>sX ( \rightarrow d \rightarrow eX )</td>
</tr>
<tr>
<td>Stamp [P] ≥ t</td>
<td>( t \rightarrow P )</td>
</tr>
<tr>
<td>Stamp [P] ≤ t</td>
<td>( P \rightarrow t )</td>
</tr>
<tr>
<td>Length [X] ≥ d</td>
<td>sX ( \rightarrow d \rightarrow P \rightarrow eX )</td>
</tr>
<tr>
<td>Length [X] ≤ d</td>
<td>sX ( \rightarrow eX )</td>
</tr>
</tbody>
</table>

Stamp and Length Constraints

System Architectures Laboratory
Working with Point Graphs

User specify the Temporal Constraints

Convert each Temporal Constraint to a Point Graph

Unify into a single Point Graph

Check for Consistency

Fold the Point Graph

Preprocessing

Revise the Point Graph (fast updates)

Run Scheduling Algorithms on Consistent Point Graph

Queries
- Find Stamp
- Find Length
- Find Relation
Operations on Point Graphs

Unification
- Unification I
- Unification II

Folding
- Branch Folding I
- Branch Folding II
- Branch Folding III

Join Folding I
- Join Folding II
Operations on Point Graphs

Verification

Check for Cycles

Check for Inconsistent Paths

Virtual Nodes

Add Source

Add Sink
Example

Constraints

Interval A, B, C, D, E
Length [A] = 5
Length [B] = 5
Length [C] = 5
Length [D] = 2
Length [E] = 10
A Meets B
C Meets D
C Precedes B
eE Precedes eD
Scheduling Algorithms

- Executed on Consistent Point Graph.
- Calculate Earliest Occurrence.
- Calculate Latest Occurrence.
- Identify Critical Activities.
- Identify Slacks/Floats for Non-Critical Activities.
The earliest occurrence of a node is the smallest time stamp that can be assigned to it such that its preceding nodes can still start at their earliest.
Earliest Occurrence

System Architectures Laboratory
Earliest Occurrence
Earliest Occurrence

\[ V_{in} \rightarrow sA \rightarrow eA; sB \rightarrow eB \]

\[ V_{out} \rightarrow eE \rightarrow sE \rightarrow sC \rightarrow eC; sD \rightarrow eD \]

Transition times:
- \( sA \rightarrow eA; sB : 5 \)
- \( eA; sB \rightarrow eB : 5 \)
- \( V_{in} \rightarrow sC : 5 \)
- \( sC \rightarrow eC; sD : 2 \)
- \( eC; sD \rightarrow eD : 2 \)
- \( eD \rightarrow V_{out} : 2 \)
- \( V_{out} \rightarrow eE : 10 \)
- \( eE \rightarrow sE : 10 \)
- \( sE \rightarrow sC : 5 \)

Initial states:
- \( sA, sC, sE \) start with \( E = 0 \)
Earliest Occurrence

System Architectures Laboratory
Earliest Occurrence

```
E = 0
sA --> eA; sB
5
E = 5
eA; sB --> eB
5
E = 10
sB --> eB

E = 0
sC --> eC; sD
5
E = 5
eC; sD --> eD
2
E = 10
eD --> V_out

E = 0
V_in --> sC
5
E = 0
sC --> eC; sD

E = 0
sE --> eE
10
E = 10
eE --> V_out
```

System Architectures Laboratory
Earliest Occurrence
Earliest Occurrence

System Architectures Laboratory
Earliest Occurrence

System Architectures Laboratory
Latest Occurrence

The latest occurrence of a node is the largest time stamp that can be assigned to it without increasing the completion time (minimum makespan) of the project.
Latest Occurrence

System Architectures Laboratory
Latest Occurrence

System Architectures Laboratory
Latest Occurrence

System Architectures Laboratory
Latest Occurrence

System Architectures Laboratory
Latest Occurrence

System Architectures Laboratory
For critical activities the earliest occurrence is equal to latest occurrence.
The presented methodology has been implemented in a software called Temper.
Questions?

Thank You For Your Attention!